VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vývoj podnikových aplikací pro iOS

Enterprise Applications Development for iOS

2013 Martin Banáš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Banáš**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Vývoj podnikových aplikací pro iOS
Enterprise Applications Development for iOS

Zásady pro vypracování:

Student nastuduje možnosti implementace aplikací pro zařízeni iOS a implementuje jednoduché API pro účely elektronického publikování včetně ukázkové tvorby. Pro tyto účely může student jako základ využít existující framework.

1. Student prozkoumá současné trendy na poli vývoje formulářových aplikací pro iOS a jejich využití pro elektronické publikování (využití Adobe technologií, HTML5, Newsstand)
2. Student navrhne a implementuje technologii pro účely automatizovaného generování obsahu pro publikace
3. Student vytvoří znovupoužitelné API pro vývoj iOS aplikací vhodných pro elektronické publikování (připojení k webovým službám a hostingu, XML, využití rozhraní Newsstand od Apple včetně implementace 'in-app' nákupů, multi-tasking atd.).
4. Student vytvoří na základě požadavků z ad 2) a API z ad 3) ukázkové aplikace vhodné pro další zobecnění pro budoucí softwarová řešení v oblasti elektronického publikování pro iOS mobilní zařízení.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. David Ježek, Ph.D.**

Datum zadání: 18.11.2011

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

… … … … … … … … … … … … …

Martin Banáš

Datum odevzdání bakalářské práce: 24. 4. 2013

Poděkování

Tímto bych rád poděkoval svému vedoucímu práce panu Ing. Davidovi Ježkovi, Ph.D. za jeho cenné rady, odborné vedení a čas, jenž mi při konzultacích věnoval. Dále patří mé poděkování společnosti AstrumQ Interactive, s.r.o., a to za poskytnuté prostředky, které přispěly k pochopení problematiky vývoje aplikací pro iOS a OS X a díky nimž došlo k úspěšnému vytvoření realizovaného ukázkového softwaru.

Abstrakt

Cílem této bakalářské práce je navrhnout a následně implementovat vlastní řešení elektronického publikování, a to na základě předchozího prozkoumání současných trendů na poli vývoje nativních aplikací pro iOS a jejich využití pro elektronické publikování. Součástí daného řešení bude technologie automatizovaného generování obsahu publikací a znovupoužitelné API pro vývoj iOS aplikací. Závěr této práce poté poskytuje shrnutí dosažených výsledků a náměty na další postup ve vývoji.

Klíčová slova

Elektronické publikování, Kiosek, iOS, OS X, iPad, iPhone, Objective-C, mobilní zařízení

Abstract

The goal of this graduation project is to design and implement a digital publishing solution based on a research of current trends of developing native iOS applications and their use in digital publishing. The solution will include a technology for automated content generation and reusable API for further development of iOS applications. The conclusion sections provides summary of the results and suggestions for further development.

Key words

Digital publishing, Newsstand, iOS, OS X, iPad, iPhone, Objective-C, mobile devices

Seznam použitých symbolů a zkratek

API	–	(angl. Application Programming Interface) programovací rozhraní aplikace
APNs	–	(angl. Apple Push Notification service) služba pro rozeslání notifikací
ARC	–	(angl. Automatic Reference Counting) automatické počítání referencí
CMYK	–	(angl. Cyan, Magenta, Yellow, Black) azurová, purpurová, žlutá, černá
CPU	–	(angl. Central Processing Unit) centrální procesorová jednotka
DPS	–	(angl. Digital Publishing Suite) komplet digitálního publikování Adobe
DTP	–	(angl. Desktop Publishing) stolní ediční systém
GPS	–	(angl. Global Positioning System) globální polohový systém
GUI	–	(angl. Graphical user Interface) grafické uživatelské rozhraní
HTML5	–	(angl. Hypertext Markup Language 5) hypertextový značkovací jazyk
HTTP	–	(angl. Hypertext Transfer Protocol) hypertextový přenosový protokol
ID	–	identifikační klíč
IDE	–	(angl. Integrated Development Environment) vývojové prostředí
JSON	–	(angl. JavaScript Object Notation) javascript-objektový zápis
MVC	–	(angl. Model View Controller) návrhový vzor
OS	–	operační systém
PDF	–	(angl. Portable Document Format) přenosný formát dokumentu
RAM	–	(angl. Random Access Memory) paměť s náhodným přístupem
RGB	–	(angl. Red, Green, Blue) červená, zelená, modrá
UI	–	(angl. User Interface) uživatelské rozhraní
XML	–	(angl. Extensible Markup Language) rozšiřitelný značkovací jazyk
ZIP	–	komprimovaný soubor

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Mobilní zařízení hrají v současné době významnou roli a jdou ruku v ruce s dnes všudypřítomnou digitalizací. Stávají se především nástrojem k oslovení co nejširší škály uživatelů a mnohdy také slouží k budování samotných značek společností. Čím dál častěji se lze setkávat se skutečností, že společnosti, které se dříve o mobilní platformy vůbec nezajímali, se už dnes bez těchto technologií de facto neobejdou. Zdárným příkladem je elektronické publikování, kdy se vydavatelství nebo jednotlivé redakce stále více zaměřují na elektronickou distribuci.

Tématu elektronického publikování se věnuje i tato bakalářská práce. V první části budou popsány nejčastější způsoby využití platformy iOS k elektronickému publikování a zároveň budou prozkoumány již existující dostupná řešení, které poslouží jako určitý zdroj inspirace.

Dále budou definovány požadavky na implementaci vlastního ukázkového řešení tak, aby co nejlépe vyvážily přednosti a nedostatky již popsaných způsobů publikování. Řešení bude rozděleno do několika částí, u kterých nebude chybět popis problémů, jež řeší. Další kapitoly poskytnou návrh realizovaných částí a podrobnější popis jejich implementace.

V závěru této práce budou shrnuty dosažené výsledky vlastního řešení a také nastíněny další možnosti rozšíření a postup vývoje.

2 Elektronické publikování na iOS

Tato kapitola se zabývá současnými způsoby elektronického publikování na platformě iOS, a dále pak popisuje některá, již dostupná řešení elektronického publikování. Kapitola shrnuje jejich přednosti a nedostatky a zároveň slouží jako inspirace pro implementaci vlastního řešení.

2.1 Newsstand aplikace

S aktualizací operačního systému iOS 5 přišla společnost Apple s novou funkcionalitou zvanou Newsstand a NewsstandKit framework. Newsstand je speciální typ obchodu, kde uživatelé naleznou aplikace pouze zaměřené na elektronické publikace. Dá se říci, že se jedná o „mini verzi App Storu“ pouze pro vydavatele. Newsstand poskytuje společné centrum pro všechny aplikace, které jsou orientovány čistě na časopisy a noviny. Apple v žádném případě nediktuje podmínky, jak takové aplikace mají vypadat a jakým způsobem mají svůj obsah prezentovat. Apple také momentálně nenabízí žádné vlastní řešení pro elektronické publikování, a tak je zcela na programátorovi, aby realizoval vlastní řešení.

Obrázek 1: Newsstand na iOS zařízeních

Jedna ze zajímavějších vlastností Newsstand aplikací, je možnost přijímat notifikace ve chvíli, kdy vydavatel má připraven nový obsah ke stažení. To znamená, že aplikace je schopna automaticky na pozadí stáhnout nový obsah a aktualizovat ikonu aplikace v Newsstandu na přední stranu nejnovější publikace. Uživatel tedy může mít k dispozici vždy aktuální číslo svého oblíbeného časopisu, a to aniž by byla vyžadována jeho pozornost. Z pohledu vydavatele přináší Newsstand marketingovou výhodu, jelikož se prezentuje jako zcela samostatná část, jenž je jednoduše viditelná a dobře podporovaná ze strany Applu. Pro uživatele je mnohem přirozenější si vybírat publikace z Newsstandu, zvláště pak, když ikony aplikací jsou aktualizovány a reprezentují právě nejnovější vydání. Nakonec z pohledu programátora, NewsstandKit framework nabízí mnoho skvělých vlastností, včetně efektivního systému stahování obsahu na pozadí a řeší sám velkou část funkcionality aplikace [1].

Apple vyžaduje v každé Newsstand aplikaci implementaci In-App nákupů a možnost si předplatit obsah i tehdy, kdy je aplikace zcela zdarma a to včetně i jejich publikací, čímž přidává programátorovi někdy zbytečnou práci navíc.

2.2 Stand-alone aplikace

V době, kdy ještě nebyl k dispozici Newsstand a NewsstandKit framework, se museli programátoři a vydavatelé spolehnout sami na sebe, vytvářet aplikace od základů a prezentovat je vlastním způsobem. I dnes existují aplikace, které nevyužívají prvky Newsstandu a mnohdy ho dokonce i zcela nahrazují. Řeč je například o velmi úspěšné aplikaci Zinio od společnosti Zinio LLC, která sama sebe prezentuje jako „Newsstand“. Aplikace Zinio[1] nepředstavuje pouze jeden časopis, ale slouží právě jako centrum všech publikovaných časopisů skrze řešení této společnosti a umožňuje jednoduše procházet řazené kategorie a jednotlivé publikace.

Obrázek 2: Stand-alone aplikace Zinio

V případě stand-alone aplikace je ale potřeba navrhnout a implementovat vše od základů a tudíž se nelze spoléhat na předpřipravenou funkcionalitu NewsstandKit frameworku. Navíc nedostatkem je určitě také stižené prezentování, neboť se aplikace velmi jednoduše ztratí mezi tisíci jinými.

2.3 Webové aplikace

Webové aplikace jsou v prvé řadě multiplatformní a poskytují konzistentní vzhled napříč všemi druhy mobilních zařízení. Tedy u nativních aplikací je řešení limitováné například pouze na platformu iOS, ale díky jazyku HTML5 lze oslovit uživatele i na zařízeních jako Android, Windows Phone a jiných. Další nespornou výhodou webových aplikací je, že odstraňují nutnost kontroly ze strany Applu před jejich samotným vydáním. Kontrola nativní iOS aplikace může trvat i několik týdnů, zatím co webová aplikace může být aktualizována téměř okamžitě, a to bez dalších zdržení.

1 Stránky Zinio jsou dostupné na adrese http://www.zinio.com

Obrázek 3: Financial Times HTML5 aplikace

Mezi hlavní nedostatky webových aplikací je jejich rychlost a výkon. Prostředky prohlížeče na zařízení jsou výrazně limitovány. Webová aplikace sice může pracovat bez neustálého připojení k internetu, nelze však skrze ni už lokálně ukládat data. Navíc nativní aplikace má přístup i k dalším funkcionalitám zařízení, jako je například GPS, mikrofon a jiné [2].

2.4 Zástupci dostupných řešení

Jak je již uvedeno výše, společnost Apple nenabízí žádné vlastní řešení pro elektronické publikování. Lze však využít řešení třetích stran, které by odstranily nutnost vyvíjet vlastní aplikaci.

2.4.1 Adobe Digital Publishing Suite

Digital Publishing Suite[2] (DPS) společnosti Adobe poskytuje vydavatelům sadu služeb a nástrojů k vytváření a publikování elektronických časopisů. Základem tvorby je DTP program InDesign, skrze který se tvoří tzv. folio, reprezentující jednu publikaci. Součástí DPS je také App Builder, který slouží k vytváření Newsstand nebo stand-alone aplikací pro zobrazování jednotlivých folií. Samotná mobilní aplikace pak nabízí jednoduchý a intuitivní design. Poutavý je také způsob stahování publikací, kdy časopis nebo noviny jsou stahovány po stránkách a uživatel si je může prohlížet ještě před tím než bude stahování publikace zcela kompletní. Nevýhodou však zůstává nutnost využívat program InDesing při tvorbě folií. Nelze tak využít přímo PDF dokumenty, které by ušetřily čas při vytváření publikace.

2.4.2 CoverPage

CoverPage[3] společnosti MONOGRAM Interactive LLC je dalším poskytovatelem služeb a nástrojů pro elektronické publikování. I CoverPage disponuje vlastním DTP nástrojem pro tvorbu

2 Adobe Digital Publishing Suite je dostupný na adrese http://www.adobe.com/cz/products/digital-publishing-suite-family.html
3 CovePage je dostupný na adrese http://www.coverpageapp.com

elektronických publikací a umožňuje podobným způsobem vytvářet nativní aplikace jako DPS. Vstupními daty může být PDF dokument, díky kterému lze velice jednoduše a rychle připravit novou publikaci k distribuci. CoverPage především vyniká interaktivními objekty, kterými lze publikace obohatit. Mezi standardní prvky dnes patří vnořená videa, vnořené webové stránky, galerie obrázků a animace. CoverPage ale zachází mnohem dále a využívá rovněž akcelerometr, gyroskop nebo GPS mobilního zařízení. Slabší stránkou zůstává renderování PDF dokumentu přímo v mobilní aplikaci, které nepatří k nejrychlejším, nicméně je i přesto dostačující.

2.4.3 Triobo

Triobo[4] je zastupována společností EDANEO s.r.o. a je aktuálně nejmladším řešením na českém trhu. Vychází ze stejného modelu publikování jako DPS a CoverPage. Skrze DTP program se připravují elektronické publikace a lze automatizovaně vytvářet nativní aplikace pro mobilní zařízení. V čem se ale od svých konkurentů výrazně liší, je forma obsahu prezentována v aplikacích. Triobo staví své řešení na jazyce HTML5. Většina vydavatelů je dnes stále orientována především na tiskovou podobu svých publikací a elektronické publikování zatím není hlavním kanálem distribuce. Ve chvíli, kdy se vydavatel rozhodne využít elektronického publikování, hledá určitě ten nejjednodušší a nejlevnější způsob. Tím je právě PDF dokument používán k tisku, který lze velice jednoduše využít i pro tvorbu elektronické publikace [3]. Triobo označuje tento způsob jako nedostatečné využití potenciálu mobilních zařízení a uvádí že elektronická publikace by neměla být pouhou kopií tištěné formy.

V případě, že se vydavatel rozhodne využít tedy právě tohoto řešení, musí počítat s tím, že publikace lze vytvářet pouze skrze Triobo DTP program za pomocí jazyku HTML5, a že takto vytvořenou publikaci nelze využít už jiným způsobem.

2.5 Shrnutí

Zdá se, že Newsstand nabízí hned několik výhod pro digitální publikování. Avšak zaměřím-li se ještě jednou na jeho nedostatky, pak musím zmínit právě ten největší. Za každý zakoupený obsah v aplikaci skrze In-App nákupy strhává společnost Apple 30% z ceny [4]. To vede některé vydavatele k tomu, že se raději obrací, na nikoli nativní aplikace pro iOS, ale na webové aplikace založené na jazyce HTML5, u kterých není strhávání poplatků za každý nákup. Co si tito vydavatelé možná ale neuvědomují je, že Newsstand je zároveň silný marketingový nástroj. Společnost Apple bez jakýchkoliv dalších nákladů celosvětově prezentuje jednotlivé publikace a umožňuje tak i menším vydavatelům se úspěšně prosadit. Proto lze říci, že poplatek za nákup v aplikaci, je vzhledem k výhodám, zřejmě akceptovatelný.

Výše uvedení zástupci elektronického publikování nabízí víceméně srovnatelná řešení a to i za podobných cenových podmínek (viz Tabulka 1). Rozhodujícím faktorem pak můžou být vstupní data při tvorbě publikace a zde se zdá nejjednodušší a nejvýhodnější využít PDF dokument.

4 Triobo je dostupné na adrese http://www.triobo.com

Řešení	Aplikace výhradně s jednou publikací	Stand-alone / Newsstand aplikace	Provize z prodeje
DPS	9 325,- Kč	9 123,- Kč / měsíčně	žádné
CoverPage	12 840,- Kč	4 883,- Kč / měsíčně	žádné
Triobo	9 990,- Kč	990,- Kč / měsíčně	35%

Tabulka 1: Cenové podmínky zástupců dostupných řešení včetně DPH

Význam mají také použité analytické nástroje jako jsou Google Analytics[5] nebo Flurry[6], které shromažďují informace o používání mobilních aplikací. Tyto nástroje umožňují rovněž vydavatelům lépe připravit obsah publikací a zaměřit se na konkrétní skupiny uživatelů.

5 Google Analytics jsou dostupné na adrese http://www.google.com/analytics
6 Flurry je dostupný na adrese http://www.flurry.com

3 Požadavky na vlastní řešení

Prvním požadavkem na vlastní řešení elektronického publikování je navrhnout celý proces tak, aby jeho uživatel nepotřeboval žádné znalosti z programování a skrze jednoduché GUI mohl automatizovaně generovat nové nativní aplikace a jejich obsah. Ze zadání této práce také vyplývá, že řešení se výhradně soustředí na mobilní zařízení s operačním systémem iOS. Řešení se bude dělit na dvě části:

- **iOS aplikace**
- **generování iOS aplikací a jejich obsahu**

3.1 iOS aplikace

Za účelem realizace iOS aplikace bude navržen a implementován znovupoužitelný framework KiosekKit, který bude podstatou celé aplikace. Bude řízen za pomoci své API. Aplikace bude nabízet elektronické publikace ke stažení nebo i zakoupení a potřebné informace bude získávat skrze internetové připojení k webovému serveru. Aplikace bude schopna uživateli plnohodnotně zobrazit obsah publikace a zároveň bude plně využívat funkcionality NewsstandKit frameworku. Aplikace by tedy měla být schopna:

- přijímat notifikace o nových publikacích
- stahovat publikace a uložit je lokálně do mobilního zařízení
- stahovat publikace v pozadí
- obnovit přerušená stahování
- odstranit už stažené publikace a uvolnit tak místo na mobilním zařízení

Apple vyžaduje v každé Newsstand aplikaci implementaci In-App nákupů a možnost si předplatit obsah. Řešení aplikace bude tedy zahrnovat i In-App nákupy za pomoci StoreKit frameworku a možnost si předplatit nové publikace.

3.2 Generování iOS aplikací a jejich obsahu

Jelikož je při generování nové iOS aplikace zapotřebí kompilace jejího zdrojového kódu, který lze kompilovat pouze na zařízeních podporující operační systém OS X, bude část, která je zodpovědná za generování nové iOS aplikace, realizována jako OS X desktopová aplikace. Tato aplikace nabídne jednoduché GUI, které provede uživatele skrze celý proces získávání potřebných informací a podkladů pro novou iOS aplikaci. Další podstatnou částí OS X aplikace bude generování nových elektronických publikací, což zahrnuje implementaci rozhraní pro uživatele, kde lze:

- vytvořit novou publikaci
- přesunout publikaci na webový server

- informovat iOS aplikaci notifikací o nové publikaci
- odstranit publikaci z webového serveru

Protože iOS aplikace mohou být distribuovány koncovým uživatelům skrze App Store společnosti Apple, je důležité aby navrhnuté řešení vždy generovalo iOS aplikaci, která bude splňovat následující podmínky:

- aplikace musí splňovat podmínky distribuce skrze App Store[7]
- chod a výkon aplikace musí být optimalizován
- aplikace musí nabídnout přijatelné GUI

3.3 Obsah publikace

Dalším požadavkem je použití vhodných vstupních dat při vytváření obsahu publikace. Důraz je kladen na co nejjednodušší a nejrychlejší způsob publikování. Vzhledem k rozboru současných způsobů publikování[8] se jeví jako nejvhodnější použití PDF dokument, který je dnes jeden z nejčastěji používaných souborových formátů pro tisk magazínů a novin. PDF dokument je nezávislý na softwaru a hardwaru, na kterém byl vytvořen. Pro tisk se však nejčastěji využívá barevných model CMYK. Je možno tak předpokládat, že většina PDF dokumentů, které byly vytvořeny za účelem tisku, využívá právě tento model [3]. iOS zařízení jsou orientována především na barevný model RGB, čímž může docházet k chybnému zobrazování barev na samotném zařízení. Proto před tím než bude PDF dokument použit jako obsah publikace, měl by být jeho barevný model změněn na RGB.

3.4 Webový server

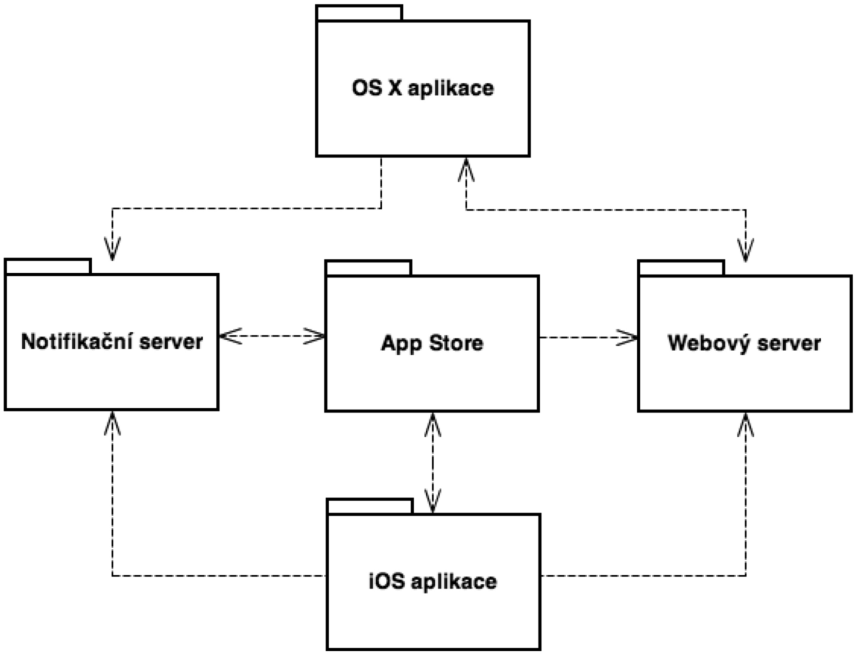
Řešení elektronického publikování vyžaduje webový server, kde OS X aplikace bude ukládat elektronické publikace a iOS aplikace bude jak k těmto publikacím, tak k jejich informacím bezprostředně přistupovat. Webový server bude následně nahrazen již existujícím řešením, jelikož jeho realizace není přímou součástí této bakalářské práce.

Cloudová služba Dropbox[9], jež je provozována společností Dropbox LLC, má k dispozici framework, který nabízí metody pro zajištění komunikace mezi API služby a koncovou iOS nebo OS X aplikací. iOS aplikace nevyžaduje přímo zajištění komunikace s webovým serverem, a proto je framework vyžadován pouze u OS X aplikace. Využití Dropboxu výrazně zjednoduší celou realizaci vlastního řešení.

7 Podmínky distribuce iOS aplikace skrze App Store jsou dostupné na adrese https://developer.apple.com/appstore/resources/approval/guidelines.html
8 Viz kapitola 2
9 Cloudová služba Dropbox je dostupná na adrese https://www.dropbox.com

4 Architektura vlastního řešení

Celé řešení elektronického publikování je rozděleno do několika komponent, kde každá komponenta je zodpovědná za specifickou funkcionalitu. Jejich propojení je omezeno na minimum, aby nedocházelo k nadměrným závislostem, které jsou nežádoucí. Rozdělení a vzájemné závislosti lze vidět na Obrázku 4.



Obrázek 4: Architektura vlastního řešení

4.1 App Store

Hlavním úkolem App Store je zajišťovat vše nezbytné pro realizaci In-App nákupů. Především se jedná o přijímání a zpracovávání žádostí z iOS aplikace za účelem nákupů elektronických časopisů. Dále pak App Store přijímá registrace jednotlivých mobilních zařízení k APNs.

4.2 Webový server

Webový server v zásadě slouží jako úložiště. OS X aplikace vkládá informace o distribuovaných publikacích na tento server a zároveň na něj ukládá i jejich obsah. iOS aplikace pak tyto informace stahuje a zobrazuje uživateli. Jak už bylo uvedeno výše[10], webový server je v tomto případě nahrazen cloudovou službou Dropbox.

10 Viz kapitola 3.4

4.3 Notifikační server

Notifikační server je vyžadován pro zasílání notifikací na jednotlivá mobilní zařízení. Ve chvíli kdy se iOS aplikace spouští, současně se autentizuje k notifikačnímu serveru. Tento server spravuje seznam mobilních zařízení, kterým lze notifikace zasílat. Ve chvíli, kdy OS X aplikace má k dispozici novou publikaci, informuje notifikační server o této situaci. Server pak následně rozešle notifikace na všechny autentizované mobilní zařízení skrze API APNs.

Realizace vlastního notifikačního serveru není součástí zadání této práce a jeho implementace by vyžadovala značné množství času. Server je tedy nahrazen claudovou službou Parse[11], která nabízí možnost rozesílání notifikací na mobilní zařízení a zároveň poskytuje API a framework, které jsou zapotřebí pro komunikaci s OS X a iOS aplikací.

4.4 OS X

OS X aplikace má především za úkol umožnit uživateli vytvořit novou iOS aplikaci. Skrze jednoduché GUI bude aplikace žádat uživatele o všechny potřebné informace a vstupní data. Následně provede samotnou kompilaci zdrojového kódu iOS aplikace a výsledný binární soubor poskytne uživateli. OS X aplikace umožňuje generovat obsah nových publikací a jejich samotnou distribuci. Tyto publikace a jejich informace jsou ukládány na webový server. Využíváno je i API notifikačního serveru k upozorňování uživatelů na novou publikaci.

4.5 iOS aplikace

iOS aplikace primárně slouží k distribuci elektronických publikací koncovým uživatelům. Tyto publikace pak lze zakoupit, stáhnout a zobrazit je na mobilním zařízení. Informace o elektronických publikacích získává iOS aplikace z webového serveru. Zároveň komunikuje s API App Store za účelem realizace In-App nákupů a přijímání notifikací.

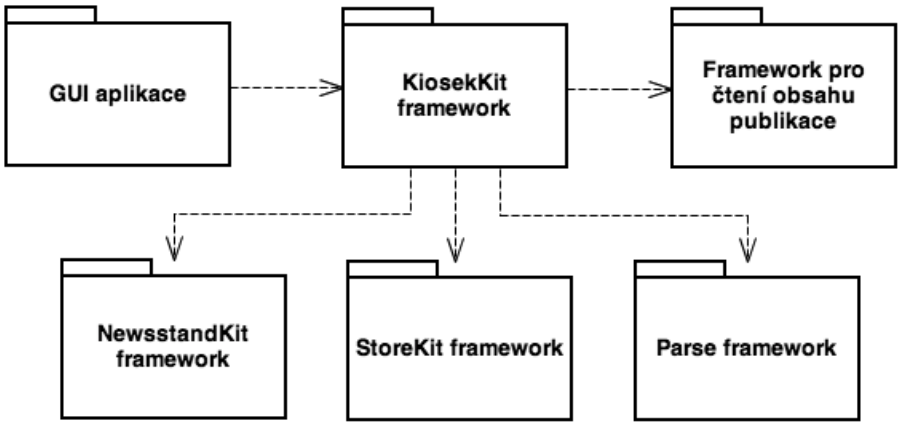
11 Claudová služba Parse je dostupná na adrese https://parse.com

5 iOS aplikace

5.1 Návrh a struktura aplikace

iOS aplikace je navržena pro mobilní zařízení iPod Touch, iPhone a iPad. Jelikož nejsou využívány žádné specifické vlastnosti (např. GPS, volání, atd.), není žádný rozdíl mezi aplikací pro iPhone nebo iPod Touch. Odlišnost lze zpozorovat až u aplikace spuštěné na iPadu, jenž disponuje daleko větším displejem. Aplikace nabízí trochu odlišné GUI, které lépe využívá větších rozměrů. To se odráží i na samotné implementaci, kdy třídy realizující GUI jsou implementovány pro iPhone a iPad zvlášť. Další popis struktury aplikace je, pro zjednodušení, redukován pouze na jedno zařízení.

Struktura iOS aplikace se dělí na tři hlavní a tři vedlejší komponenty (viz Obrázek 5). Mezi hlavní komponenty patří GUI aplikace, KiosekKit framework a framework pro čtení obsahu publikace. Mezi vedlejší komponenty patří NewsstandKit framework a StoreKit framework, jenž jsou součástí Cocoa Touch. Poslední komponentou je pak Parse framework realizující komunikaci s notifikačním serverem.



Obrázek 5: Struktura iOS aplikace

- **GUI aplikace** reprezentuje třídy zpracovávající veškeré události vyvolané uživatelem, rovněž také reagují na změny propagované KiosekKit frameworkem.
- **KiosekKit framework** je hlavní podstatou a spravuje funkcionalitu, kterou má mít plnohodnotná Newsstand aplikace. Zajišťuje komunikaci s API App Store v rámci In-App nákupů a notifikací. Taktéž spravuje celý proces stahování, ukládání a zobrazování obsahu elektronických publikací.
- **Framework pro čtení obsahu publikací** je další částí iOS aplikace. Jak už bylo uvedeno výše[12], obsah publikace je reprezentován PDF dokumentem. Lze tedy využít jakýkoliv framework umožňující uživateli zobrazit PDF dokument na mobilním zařízení.

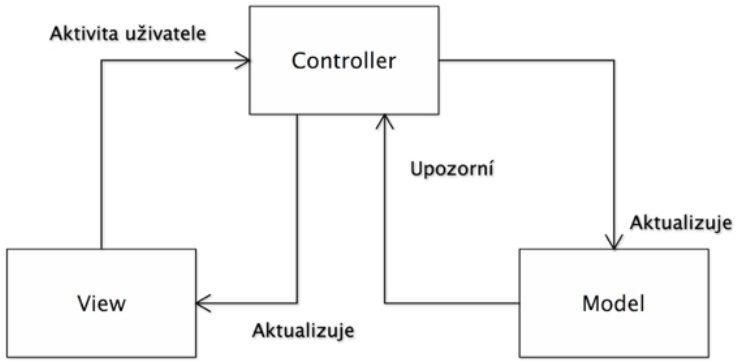
12 Viz kapitola 3.3

- **NewsstandKit framework** poskytuje sadu tříd, jež implementují metody pro asynchronní stahování publikací a jejich správu na mobilním zařízení. Více lze nalézt v kapitole 5.5.
- **StoreKit framework** poskytuje třídy, které umožňují aplikaci realizovat In-App nákupy. Více lze nalézt v kapitole 5.6.
- **Parse framework** implementuje metody pro autentizaci mobilního zařízení k notifikačnímu serveru, a to aby bylo možné zasílat notifikace na mobilní zařízení.

5.2 Hlavní návrhové vzory

5.2.1 MVC

Model-View-Controller (MVC) je jeden z nejstarších a nejpoužívanějších návrhových vzorů Cocoa a Cocoa Touch. Model zapouzdřuje data aplikace, view data zobrazuje a upravuje. Controller následně zprostředkovává komunikaci mezi modelem a view. Tímto rozdělením lze aplikaci mnohem jednodušeji navrhnout a implementovat.



Obrázek 6: Návrhový vzor MVC

Další výhodou MVC je možnost využít Cocoa bindings. Cocoa bindings je sada technologií zjednodušujících implementaci MVC. Tyto technologie redukují množství kódu, které jsou zapotřebí mezi modelem, view a controllerem a zároveň automaticky aktualizují view při změně v modelu. Cocoa bindings lze však použít pouze u implementace OS X aplikace, jelikož je součástí Cocoa a nikoli už Cocoa Touch [5].

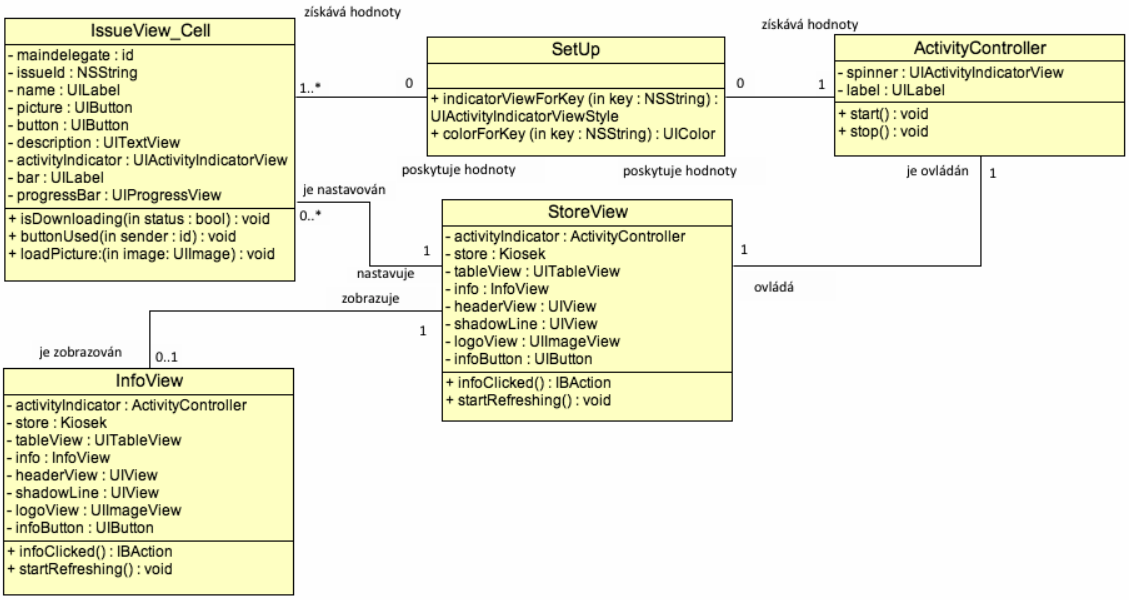
5.2.2 Delegát

Delegát je velmi jednoduchým a jedním z nejčastěji používaných návrhových vzorů při realizaci obou aplikací. Zpravidla zajišťuje synchronní nebo asynchronní komunikaci mezi objekty. Jeden objekt si udržuje referenci na druhý objekt a ve vhodnou chvíli informuje delegáta, že v druhém objektu nastala konkrétní událost. Hlavním přínosem delegáta je skutečnost, že umožní snadno spravovat komunikaci množiny objektů v jednom centrálním objektu [6].

5.3 Implementace aplikace

5.3.1 Uživatelské rozhraní

Třídní diagram GUI zobrazuje hlavní třídy a jejich vztahy (viz Obrázek 7).



Obrázek 7: Konceptuální datový model tříd GUI

Vzhledem k tomu, že GUI aplikace není přímo propojeno s KiosekKit frameworkem a jen zobrazuje informace, jenž díky němu získá, lze jej přizpůsobit dle potřeby a dispozicím zařízení. Aplikace spuštěná na iPhonu zobrazuje jednotlivé publikace jako vertikálně orientovaný seznam za pomoci třídy `UITableView`. Naopak aplikace spuštěná na iPadu zobrazuje publikace v mřížce pomocí třídy `UICollectionView`. Lze tedy implementovat vhodnější způsob. Dále je pak možné rozšířit už existující GUI o další prvky, mezi které mohou patřit např. animace, různé přechody, náhledy, odlišně indikátory apod.

Třída	**Její úloha**
`StoreView`	Spravuje seznam dostupných elektronických publikací a inicializuje další UI prvky. Komunikuje přímo s API KiosekKit frameworku.
`IssueView_Cell`	Reprezentuje buňku v seznamu dostupných elektronických publikací a zobrazuje informace o konkrétní publikaci.
`InfoView`	Reprezentuje pohled s informacemi o aplikaci.
`ActivityController`	Reprezentuje ukazatel zaneprázdnění aplikace.
`SetUp`	Umožňuje ostatním UI prvkům získat informace z nastavení aplikace a přizpůsobit tak jejich vzhled. Např. uživatel při generování aplikace nastavil, aby se název publikace zobrazoval modře.

	Následně si UILabel, který reprezentuje název publikace, při své inicializaci vyžádá barvu právě z entity SetUp.

Tabulka 2: Seznam tříd realizující GUI iOS aplikace

5.3.2 Konfigurace iOS aplikace

Jelikož aplikace vyžaduje informace, odkud má např. získávat seznam dostupných publikací, jaké předplatné je uživateli nabízeno, jak ve skutečnosti mají vypadat UI prvky atd., je v aplikaci vytvořen speciální konfigurační soubor `Settings.plist` obsahující tyto informace. Tento soubor je vytvořen při generování nové iOS aplikace a po celou dobu jejího užívání zůstává nezměněn. Property list (plist) je soubor, který se primárně využívá pro ukládání serializovaných objektů na operačních systémech iOS a OS X. Takový soubor dokáže přečíst i člověk a je zároveň založen na standardech formátu XML [7].

Aby iOS aplikace nemusela neustále přistupovat ke konfiguračnímu souboru, je tento soubor synchronizován se speciální komponentou `NSUserDefaults`. Tato komponenta poskytuje rozhraní pro interakci s výchozím nastavením. Umožňuje přistupovat k výchozímu nastavení odkudkoli v aplikaci a to i při přístupu z více vláken. Výchozí hodnoty a informace ukládá v paměti aplikace, čímž odpadá nutnost neustálého čtení ze skutečného souboru.

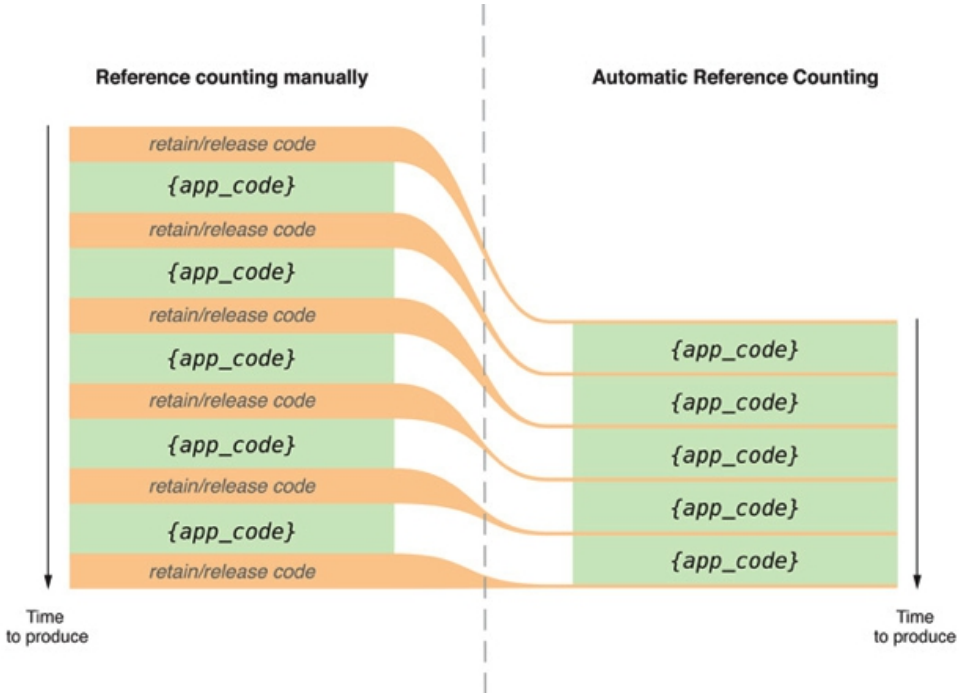
```
[[NSUserDefaults standardUserDefaults] registerDefaults:
[NSDictionary dictionaryWithContentsOfFile:[[NSBundle mainBundle]
pathForResource:@"Settings" ofType:@"plist"]]];
```

Výpis 1: Synchronizace konfiguračního souboru Settings.plist s komponentou NSUserDefaults

5.3.3 Využití ARC

Automatic Reference Counting (ARC) je jedna z novějších technologií kompilátoru Xcode. ARC poskytuje automatickou správu paměti pro objekty v jazyce Objective-C. Ve velké míře usnadňuje práci a výrazně snižuje šanci, že program bude obsahovat úniky paměti. Místo toho, aby bylo potřeba si pamatovat, kdy použít příkazy jako např. `retain` a `release`, ARC sám vyhodnocuje nejlepší postup při správě objektů v paměti a tyto příkazy vloží automaticky přímo při kompilaci kódu. ARC nepřináší nový způsob jak pracovat s pamětí v jazyce Objective-C, ale vychází ze stejné konvence jako ruční počítání referencí. Při kompilaci přidává pouze kód, který je potřebný pro vhodnou správu objektů v paměti [8].

V době, kdy začala realizace této práce, ARC ještě nebylo k dispozici. Bylo tedy zapotřebí využívat manuální počítání referencí. Ve chvíli, kdy implementace iOS aplikace byla téměř hotová, ARC bylo uvedeno k dispozici s nejnovější aktualizací Xcode. Značnou výhodou je, že Xcode umožňuje automaticky převést kód, ve kterém jsou reference objektů počítány manuálně. Místo toho, aby byla pozornost ve větší míře soustřeďována na správné používání příkazů `retain`, `release` a vyhledávání úniků paměti, bylo následně možné automaticky začít používat ARC a zaměřit se na důležitější úkoly.



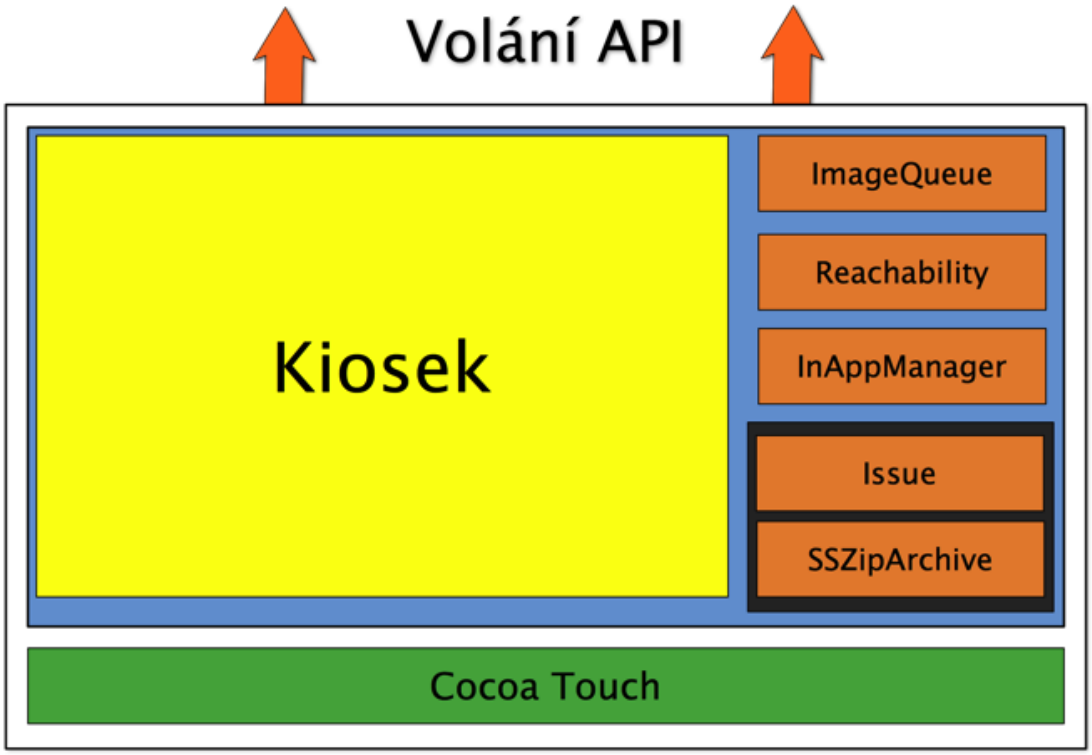
Obrázek 8: Redukce času při využití ARC

5.4 KiosekKit framework

5.4.1 Architektura KiosekKit frameworku

Poněvadž je kladen velký důraz na znovupoužitelnost tohoto frameworku je nutné, aby jeho samotné uskupení bylo dobře promyšleno. Framework a jeho jednotlivé komponenty jsou tedy navrženy tak, aby jejich využití bylo co nejjednodušší a nikterak složité k pochopení. KiosekKit je přitom zcela soběstačný a není přímo propojen s GUI aplikace. V případě zájmu lze GUI zcela zaměnit, a to aniž by bylo potřeba upravit implementaci frameworku.

KiosekKit framework se skládá ze šesti komponent tvořících podstatu Newsstand aplikace. K frameworku lze přistupovat skrze API, které je implementováno na rozhraní hlavní komponenty a její podrobnější popis je přiložen v Příloze 2.



Obrázek 9: Architektura KiosekKit frameworku

- **`Kiosek`** inicializuje a přímo ovládá ostatní komponenty frameworku[13]. Jeho implementace představuje převážnou část funkcionality.
- **`ImageQueue`** slouží pro asynchronní stahování předních stran publikací. Ve chvíli, kdy `Kiosek` stáhne seznam z webového serveru, začne procházet informace o jednotlivých publikacích a kontrolovat, zda je na mobilním zařízení obrázek již dostupný. Jestliže ne, `Kiosek` vytvoří instanci `ImageQueue` a předá ji potřebné parametry.
- **`Reachability`** komponenta má na starost ověřování dostupnosti internetového připojení.
- **`InAppManager`** komponenta spravuje veškerou logiku týkající se In-App nákupů.
- **`Issue`** představuje instanci elektronické publikace a umožňuje o ní získávat informace. Současně se stará o procesy stahování a ukládání .

5.4.2 Framework pro čtení obsahu

Za účelem nezávislosti na konkrétním frameworku zobrazující obsah publikace, je implementován protokol KiokekContentSource, jenž je zodpovědný za získání instance frameworku. Protokol v jazyce Objective-C se především používá ve chvíli, kdy je zapotřebí deklarovat metodu pro specifickou situaci, ale zároveň je vyžadováno zachovat nezávislost na konkrétní třídě.

13 Až na SSZipArchive, který je využíván výhradně třídou Issue

```
@protocol KiosekContentSource <NSObject>
@required
- (UIViewController *) loadContentReaderInstance:(NSURL*)kPath;
@end
```

Výpis 2: Definice protokolu KiosekContentSource

5.4.3 Načtení informací z webového serveru

Aby iOS aplikace mohla úspěšně zobrazit seznam dostupných elektronických publikací, musí nejdříve získat potřebné informace z webového serveru. Kde přesně se tyto informace nachází, aplikace ví díku konfiguračnímu souboru `Settings.plist`, který je synchronizován s komponentou `NSUserDefaults`. KiosekKit framework přístupem do `NSUserDefaults` získá přesnou adresu souboru `Store.plist` na webovém serveru a načte veškeré potřebné informace o elektronických publikacích. Informace a jejich struktura, jenž aplikace potřebuje pro zobrazení publikací, jsou podrobněji popsány v kapitole 6.4.1.

5.5 Newsstand

5.5.1 Konfigurace Newsstand aplikace

Aby operační systém rozeznal aplikaci jako Newsstand, je zapotřebí realizovat několik změn v jejím konfiguračním souboru. Jako první je přidán klíč `UINewsstandapp` s boolean hodnotou `true`, který dává vědět systému, že se jedná právě o Newsstand aplikaci. Dále je přidán klíč `UIBackgroundModes` se string hodnotou `newsstand-content`, který umožní stahovat publikace i na pozadí aniž by byla aplikace přímo aktivní. Poslední změnou je pak přidání nastavení speciální Newsstand ikony (viz Výpis 3).

```
<key>UINewsstandIcon</key>
      <dict>
            <key>CFBundleIconFiles</key>
            <array>
                  <string>newsstand_cover.png</string>
            </array>
            <key>UINewsstandBindingEdge</key>
            <string>UINewsstandBindingEdgeLeft</string>
            <key>UINewsstandBindingType</key>
            <string>UINewsstandBindingTypeMagazine</string>
      </dict>
```

Výpis 3: Nastavení Newsstand ikony v Info.plist

5.5.2 Knihovna publikací NKLibrary

Jedna z nejzajímavějších vlastností NewsstandKit frameworku je třída `NKLibrary`, která představuje speciální sdílenou knihovnu elektronických publikací na mobilním zařízení. Každá publikace je v této knihovně reprezentována instancí třídy `NKIssue`. Tato knihovna zároveň poskytuje speciální prostor, kde je ukládán obsah jednotlivých publikací. `NKLibrary` je zodpovědná za přidávání a odstraňování publikací, což zároveň znamená vytváření a odstraňování prostoru pro obsah každé publikace. Prostor pro publikace připomíná dočasnou složku, kdy v případě, že na mobilním zařízení docházejí prostředky k ukládání nových dat, je tento prostor i s obsahem odstraněn [9]. Je proto zapotřebí, aby KiosekKit vždy kontroloval, zda je obsah publikace stále k dispozici. V případě, že opravdu dojde k odstraňování obsahu z mobilního zařízení, jsou nejdříve odstraňovány publikace s nejstarším datem publikování. Aby nedošlo k odstranění obsahu publikace, kterou uživatel právě čte, KiosekKit přiřazuje instanci publikace `NKIssue` atribut `currentlyReadingIssue`.

```
// nastavi pro newsstand aktualne ctene issue
[[NKLibrary sharedLibrary] setCurrentlyReadingIssue:[[NKLibrary
sharedLibrary] issueWithName:anIssue.issueId]];
```

Výpis 4: Nastavení právě čtené publikace

Instance třídy `NKIssue` představuje instanci elektronické publikace v knihovně `NKLibrary`. Chce-li aplikace stáhnout nový obsah, musí nejdříve vytvořit instanci této třídy a předat ji unikátní ID a datum, kdy byla publikace publikována [9]. KiosekKit framework vytváří tyto instance již v okamžiku, kdy zpracovává data o publikacích stažené z webového serveru. `NKIssue` ale představuje pouze zlomek informací, které jsou aplikací vyžadovány. Proto KiosekKit implementuje třídu `Issue`, která již plnohodnotně reprezentuje elektronickou publikaci a pojímá všechny její potřebné informace.

5.5.3 Stahování publikace

`NKLibrary` spravuje seznam právě probíhajících stahování nebo-li instance třídy `NKAssetDownload`, které bezprostředně využívají speciálního protokolu pro stahování obsahu publikací. Řeč je o protokolu `NSURLConnectionDownloadDelegate`, který implementuje tyto tři metody:

- `connection:didWriteData:totalBytesWritten:expectedTotalBytes:`
  - tato metoda informuje o průběhu stahování publikace
- `connectionDidResumeDownloading:totalBytesWritten:expectedTotalBytes:`
  - tato metoda informuje o průběhu stahování po obnovení
- `connectionDidFinishDownloading:destinationURL:`
  - tato metoda slouží pro další zpracování staženého obsahu[14]

14 Například k odarchivování stažených souborů

Ve chvíli, kdy uživatel vybere publikaci, jenž bude stažena do mobilního zařízení, je zároveň získána instance `NKIssue` [9]. Na základě této instance se zahájí nová operace stahování a její správa je přiřazena odpovídající instanci třídy `Issue`, která implementuje metody protokolu `NSURLConnectionDownloadDelegate`.

```
[[NKLibrary sharedLibrary] addIssueWithName:_issueId
date:_published];
NSURLRequest *downloadRequest = [NSURLRequest requestWithURL:
[NSURL URLWithString:downloadURL]];

NKIssue *nkIssue = [issueToDownload newsstandIssue];
NKAssetDownload *assetDownload = [nkIssue
addAssetWithRequest:downloadRequest];
[assetDownload downloadWithDelegate:issueToDownload];
```

Výpis 5: Zahájení stahování publikace

Stahovaný obsah publikace je reprezentován archivovaným souborem obsahující PDF dokument. Po úspěšném stažení nechá aplikace tento soubor odarchívovat a přesunout do úložiště spravovaného třídou `NKLibrary`.

5.5.4 Stahování na pozadí

Jestliže aplikace je pozastavena[15] ve chvíli, kdy právě probíhá stahování publikace, operační systém obnoví činnost této aplikaci na pozadí a nechá stahování dokončit.

V případě, že aplikace byla rovnou ukončena[16], nastává odlišná situace. Operační systém aplikaci znovu spustí na pozadí a definuje speciální klíč[17] uvnitř parametru `launchOptions`. Současně je při inicializaci aplikace vyžadována implementace metody, která přerušená stahování obnoví [10].

```
if ( [[NKLibrary sharedLibrary] downloadingAssets].count != 0)
{
      for(NKAssetDownload *asset in [[NKLibrary sharedLibrary]
      downloadingAssets])
      {
            [_store setAssetIssue:asset.issue.name];
      }
      _store.status = StoreStatusDownloading;
      [_store downloadFinished:nil];
}
```

Výpis 6: Obnoveni přerušených publikací při spuštění aplikace

15 Uživatel např. spustí jinou aplikaci
16 Aplikace už není ani v paměti operačního systému
17 `UIApplicationLaunchOptionsNewsstandDownloadsKey`

5.5.5 Přijímání notifikací

Díky notifikacím může aplikace stahovat nejnovější publikace do mobilního zařízení a to zcela automatizovaně a na pozadí. Jakým způsobem je doručována notifikace do mobilního zařízení z notifikačního serveru, je podrobněji popsáno v kapitole 6.4.3.

Pro zasílání notifikací na mobilní zařízení poskytuje společnost Apple speciální službu Apple Push Notification service. Aby aplikace mohla přijímat notifikace, je vyžadována její registrace k APNs. Registrace se skládá z následujících kroků, kdy je při inicializaci aplikace volána metoda `registerForRemoteNotificationTypes:`, která umožní vytvořit šifrované spojení s APNs. Při inicializaci aplikace je zároveň provedena autentizace k notifikačnímu serveru, a to pro zajišťení vzájemné komunikace. Delegát aplikace dále využívá metodu `application:didRegisterForRemoteNotificationsWithDeviceToken:`, která umožní získat speciální klíč od APNs služby. Tento klíč je následně zaslán na notifikační server, který ho uloží v databázi pro situaci kdy budou rozesílány nové notifikace [10]. Komunikace s notifikačním serverem je realizována díky frameworku Parse poskytující třídy pro navázání spojení s API notifikačního serveru. Aplikace je připravena přijímat notifikace jakmile jsou všechny kroky úspěšně dokončeny.

```
// Registrace zarizeni k Push Notifications
[application
registerForRemoteNotificationTypes:UIRemoteNotificationTypeAlert|
UIRemoteNotificationTypeBadge|UIRemoteNotificationTypeSound|
UIRemoteNotificationTypeNewsstandContentAvailability];

// Pripojeni k API pro Push Notifications - Parse.com
[Parse setApplicationId:[[NSUserDefaults standardUserDefaults]
stringForKey:@"PARSE_APP_ID"] clientKey:[[NSUserDefaults
standardUserDefaults] stringForKey:@"PARSE_CLIENT_KEY"]];
```

Výpis 7: Autentizace a registrace k notifikacím

V momentě přijetí notifikace můžou nastat dva scénáře. V prvním případě je aplikace aktivní a její delegát volá metodu `application:didReceiveRemoteNotification`. Aplikace upozorní uživatele, že je k dispozici nové vydání a aktualizuje svůj seznam publikací. V druhém případě operační systém aplikaci musí nejdříve spustit a definovat speciální klíč[18] uvnitř parametru `launchOptions`, který obsahuje i data notifikace. Součástí těchto dat je ID nového vydání, který je předán KiosekKit frameworku [10]. KiosekKit framework aktualizuje seznam publikací a zahájí stahování obsahu publikace, jež odpovídá ID z dat notifikace.

5.6 Realizace In-App nákupů

Jak již bylo uvedeno výše, každá aplikace umístěna v Newsstandu je povinna podle pravidel společnosti Apple nabízet možnost zakoupit si předplatné, a to i přesto, že aplikace a její

18 `UIApplicationLaunchOptionsRemoteNotificationKey`

obsah bude zcela zdarma. V takovém případě bude i předplatné zdarma, avšak zůstává povinnost In-App nákupy implementovat. Jestliže bude publikace placená či nikoliv a jaké předplatné bude koncovému uživateli nabízeno, se rozhodne při generování nové publikace nebo při vytváření aplikace samotné.

Komunikaci mezi API App Store a KiosekKit zajišťuje StoreKit framework, skrze který aplikace získává lokalizované informace o produktech, jež lze v aplikaci zakoupit. Jedná se především o informace o konkrétních publikacích nebo předplatném.

StoreKit framework definuje hned několik druhů produktů, avšak pro realizaci této práce, postačí popsat pouze tři.

- **Non-consumable** produkt je zakoupen pouze a jenom jednou v rámci jednoho konkrétního uživatele. Po zakoupení je produkt k dispozici na všech ostatních zařízeních, které jsou asociovány k onomu uživateli. Uživatel tak nemusí za stejný produkt platit více než jednou, a to ani po opětovném stažení aplikace.
- **Auto-renewable subscription** je druh produktu, který představuje předplatné. Tak jako non-consumable ani tento produkt nelze zakoupit více než jednou a je dostupný na všech zařízeních uživatele zároveň. V čem se ale výrazně liší, je to, že tento produkt je časově omezen. Jestliže uživatel neurčil jinak, tento produkt se po vypršení platnosti automaticky znovu zakoupí.
- **Free subscription** je speciální druh produktu, který lze využít pouze u Newsstand aplikací. Je to způsob, jak uživateli nabízet předplatné zdarma. Toto předplatné není časově omezeno a je opět dostupné na všech zařízeních uživatele zároveň.

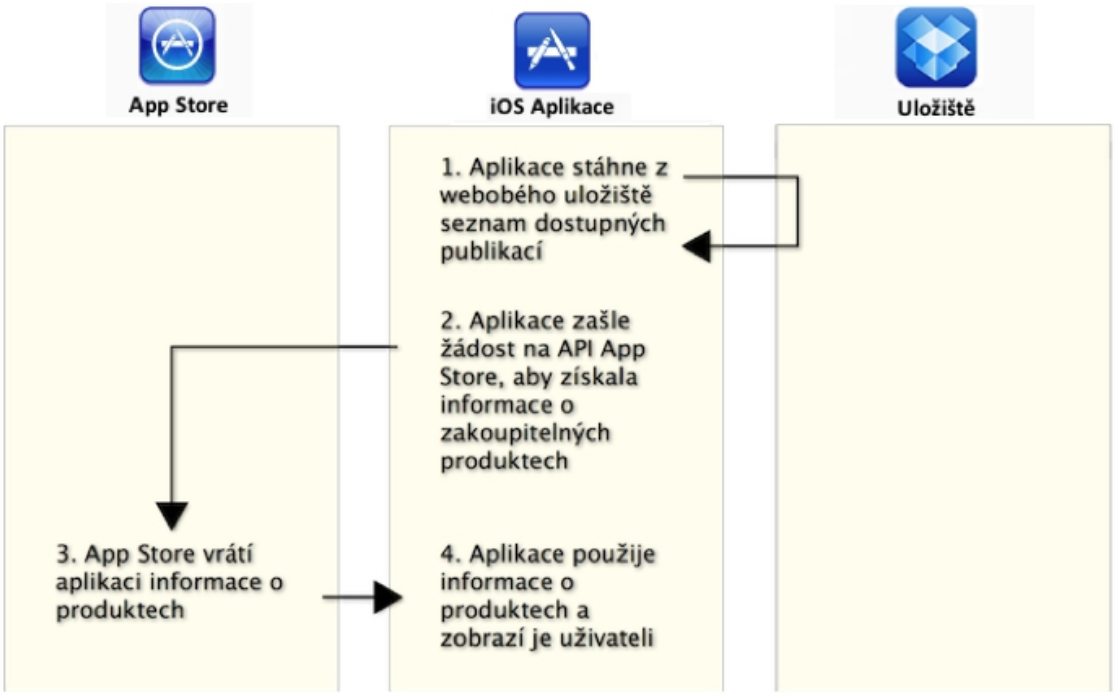
StoreKit framework zajišťuje pouze malou část implementace In-App nákupů. Je na aplikaci, jakým způsobem budou informace o produktech uživateli prezentovány, jak bude aplikace ověřovat, zda placené předplatné je stále aktivní nebo jakým způsobem bude uživateli zpřístupněn produkt, který si zakoupil. Vše ostatní tedy řeší implementace KiosekKit frameworku [11].

5.6.1 Získávání informací o produktech

Každá elektronická publikace, která je v iOS aplikaci nabízena za poplatek, je ve skutečnosti zastupována produktem, jenž lze koupit. Seznam těchto produktu je ale třeba nejdříve získat skrze App Store. Proces získání informací je znázorněn na Obrázku 10.

Na základě stažených informací KiosekKit vytvoří seznam ID zpoplatněných publikací. Následně se obrátí na třídu `InAppManager` spravující veškerou logiku týkající se In-App nákupů a předá jí tento seznam. `InAppManager` vytvoří žádost a tu následně zašle na API App Store. Ve chvíli, kdy App Store tuto žádost zpracuje, asynchronně doručí lokalizované informace[19] o produktech zpět třídě, která je předá KiosekKit frameworku za účelem předání uživateli.

19 Např. cena za časopis v lokální měně



Obrázek 10: Získání informací o zakoupitelných produktech

5.6.2 Zakoupení produktu

Ve chvíli, kdy se uživatel rozhodne zakoupit si publikaci, předá KiosekKit ID produktu[20] třídě `InAppManager`, jež vytvoří platbu a přepošle ji na API App Store.

```
SKPayment *payment = [SKPayment paymentWithProduct:product];
[[SKPaymentQueue defaultQueue] addPayment:payment];
```

Výpis 8: Vytvoření platby a její přidání do seznamu

`InAppManager` implementuje metodu `paymentQueue:updatedTransaction:` protokolu `SKPaymentTransactionObserver`, která informuje o stavu probíhající platby. Jestliže platba selže[21], KiosekKit celý proces ukončí a uživatel se může později pokusit zakoupit publikaci znovu. V případě, že platba proběhla úspěšně, je zapotřebí ji nejdříve ještě ověřit.

Když App Store vrátí úspěšnou platbu zpátky třídě `InAppManager`, platba obsahuje tzv. účtenku, ve které jsou uvedeny všechny důležité informace o právě provedené platbě. Aby bylo zabráněno manipulaci a obcházení In-App nákupů, je platba ověřována zasláním účtenky zpátky na App Store. Za účelem ověření platby je vytvořen JSON objekt obsahující její účtenku. Pomocí protokolu HTTP je objekt zaslán na API App Store, která následně odpoví, jestli je platba validní nebo nikoli. Teprve poté může být uživateli zpřístupněn zakoupený produkt [11]. Jedná-li se o produkt non-consumable, je uživateli umožněno stáhnout si konkrétní publikaci. Jestliže se jedná o

20 Id produktu se shoduje s ID publikace – získání ID produktu viz kapitola 5.6.1
21 Např. výpadkem připojení k internetu nebo nedostupnosti API App Store

předplatné, je uživateli umožněno stáhnout si poslední publikaci. V momentě, kdy bude k dispozici nové vydání publikace a zároveň předplatné bude stále aktivní, bude mu toto vydání zpřístupněno automaticky.

5.6.3 Obnovení nákupů

Aplikace využívá produktů, jenž lze zakoupit pouze jednou. Proto je důležité, aby uživatel mohl po opětovném stažení aplikace[22] či na jiném mobilním zařízení, které je asociováno s totožným uživatelským ID, zpřístupnit všechny již zakoupené publikace. Za tímto účelem je implementováno v KiosekKit frameworku metoda pro obnovení všech zakoupených produktů, a to i např. právě aktivní předplatné. Chce-li uživatel obnovit svoje nákupy, `InAppManager` zašle opět žádost na API App Store o získání seznamu všech spěšně provedených nákupů. Jakmile App Store vrátí úspěšné nákupy, je následující postup je téměř totožný s přijetím úspěšné platby za zakoupený produkt.

5.7 Testování

Veškeré části byly testovány především v samotném průběhu jejich vývoje, tedy ve fázi návrhu a poté hlavně ve fázi realizace, kdy byly postupně laděny a testovány. K testování aplikace bylo využito zařízení iPad 2 a iPhone 4s jejichž specifikace se nachází v Tabulce 3. Zvolená zařízení se ve většině parametrů shodují, leč výrazný rozdíl představují jejich displeje.

Komponenta	**Sestava použitá při testování**	**Sestava použitá při testování**
OS	iOS 6.1.3	iOS 6.1.3
CPU	Dvoujádrový Apple A5 1GHz	Dvoujádrový Apple A5 1GHz
RAM	512 MB	512 MB
Displej	768 x 1024 (132 ppi)	640 x 960 (326 ppi) Retina displej

Tabulka 3: Specifikace sestav použitých při testování iOS aplikace

Jestliže iOS aplikace není distribuována skrze App Store, lze využít prostředí sandboxu a testovat průběh In-App nákupů a notifikací. Aby bylo možné testovat zakoupení a validaci předplatného, je v rámci sandboxu jeho platnost výrazně snížena (viz Tabulka 4). Samotný sanbox je však výrazně pomalejší než reálné prostředí a při testování docházelo k velk7m prodlevám při komunikaci s APNs a App Store.

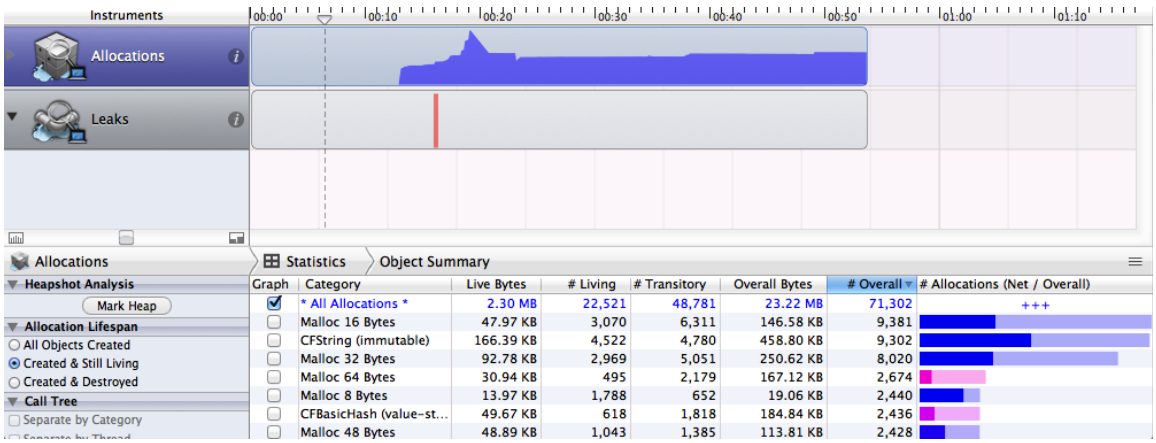
Reálná doba platnosti	**Doba platnosti v sandboxu**
1 týden	3 minuty
1 měsíc	5 minut
2 měsíce	10 minut

22 Po odstranění aplikace z mobilního zařízení, jsou vymazány i jakékoliv informace o In-App nákupech

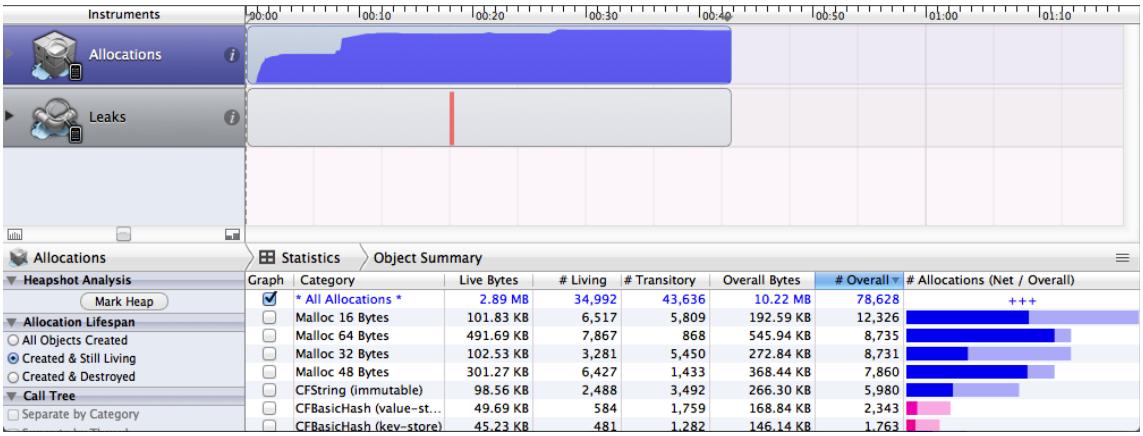
3 měsíce	15 minut
6 měsíců	30 minut
1 rok	1 hodina

Tabulka 4: Redukce času platnosti předplatného v sandboxu

Při testování probíhala především optimalizace využití paměti mobilního zařízení a lokalizace možných úniků. Aplikace nevyužívá příliš mnoho zdrojů a její reakční doba je spíše ovlivněna kvalitou internetového připojení.



Obrázek 11: Testování využití paměti na zařízení iPhone 4s



Obrázek 12: Testování využití paměti na zařízení iPad 2

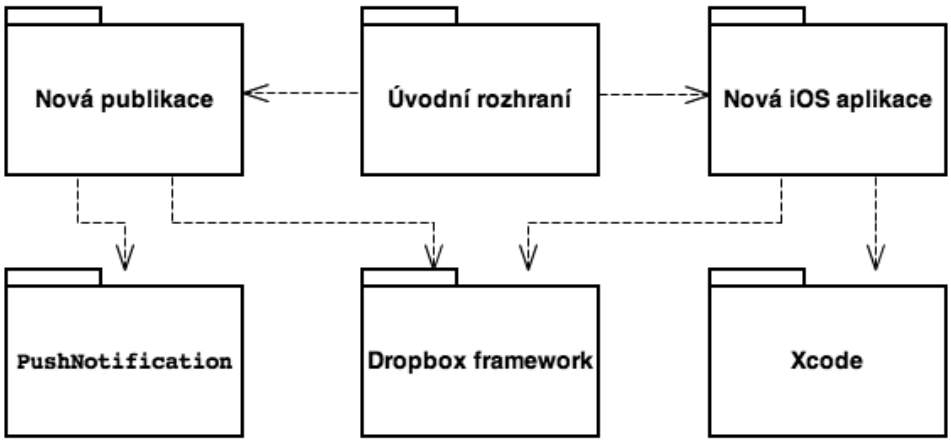
6 OS X aplikace

6.1 Návrh a struktura OS X aplikace

Hlavním úkolem OS X aplikace je vytváření nových iOS aplikací a generování publikací. Pro snadnější pochopení celé logiky je aplikace rozdělena na 3 menší komponenty:

- úvodní rozhraní
- nová iOS aplikace
- nová publikace

Aplikace také využívá dvě již dříve popsané komponenty[23]. Webový server sloužící jako úložiště informací a obsahu publikací, a notifikační server, který upozorňuje iOS aplikace na nové vydání. Komunikaci mezi webovým serverem a aplikací zajišťuje Dropbox framework, jenž implementuje nezbytné metody. Pro komunikaci mezi notifikačním serverem a aplikací postačí implementovaná třída `PushNotification` zasílající žádosti na server pomocí protokolu HTTP. Poslední a zatím nezmíněnou komponentou aplikace je IDE program Xcode, který je zodpovědný za kompilaci zdrojového kódu šablony iOS aplikace. Rozdělení aplikace je znázorněno na Obrázku 13.



Obrázek 13: Struktura OS X aplikace

- **Úvodní rozhraní** umožňuje uživateli nejen procházet již vytvořené iOS aplikace, ale rovněž je odstranit nebo začít vytvářet zcela nové. Dále pak lze spravovat seznam publikací každé iOS aplikace, kde uživatel může generovat nové publikace a aktualizovat nebo odstranit již vytvořené.
- **Nová iOS aplikace** poskytuje uživateli jednoduché GUI, které ho provede celým procesem

23 Viz kapitola 4

shromažďování potřebných informací a grafických materiálů. Na závěr je aplikována šablona iOS aplikace. Za pomoci Xcode je její zdrojový kód kompilován do konečného binárního souboru.

- **Nová publikace** je částí umožňující generovat nové publikace. Po shromáždění nezbytných informací je pomocí Dropbox frameworku nová publikace uložena na webový server, odkud ji lze stáhnout do iOS aplikace.
- **Dropbox framework** poskytuje třídy zajišťující komunikaci mezi aplikací a webovým serverem.
- **Třída PushNotification** je primárně zodpovědná za zasílání žádostí o rozeslání notifikací na notifikační server, a to v momentě, kdy je k dispozici nová publikace.

6.2 Úvodní rozhraní

Úvodní rozhraní je první část UI, která je uživateli při spuštění aplikace zobrazena. Při jeho první inicializaci je vytvořena na disku podpůrná složka `~/Library/Application Support/MBKit App/`, jež uchovává dočasné i trvalé soubory obsahující informace o iOS aplikacích a jejich publikacích. Dále pak úvodní rozhraní umožňuje uživateli procházet již vytvořené iOS aplikace nebo je odstranit či začít vytvářet zcela nové. Poslední částí je seznam publikací každé iOS aplikace, kde uživatel může generovat nové, aktualizovat nebo odstraňovat již vytvořené publikace.

6.3 Nová iOS aplikace

Vytváření nové iOS aplikace pojímá 3 důležité části. V následujících podkapitolách budou podrobněji popsány a definovány jejich jednotlivé procesy.

6.3.1 Využití služby Dropbox

Při vytváření nové iOS aplikace je potřeba předem definovat, odkud budou informace o dostupných publikacích stahovány. Tyto informace jsou ukládány na úložiště konkrétního uživatelského účtu služby Dropbox. Před tím, než OS X aplikace bude moci k tomuto úložišti přistupovat, je zapotřebí provést její asociaci.

Vývojářský portál služby Dropbox[24] umožňuje definovat aplikaci, která bude přistupovat k uživatelskému účtu. Tento portál poskytne dva klíče, jenž jsou vyžadovány při navázání spojení s API a definuje přístupová práva samotné aplikaci. Za účelem asociace OS X aplikace k uživatelskému účtu, je při její inicializaci vytvářena instance třídy `DBSession`, kterou poskytuje Dropbox framework. Následným zavoláním metody `authenticate`, je vyžádán přístup ke konkrétnímu uživatelskému účtu.

```
DBSession* dbSession = [[DBSession alloc]
initWithAppKey:@"r1ah7tfgaq4h31c" appSecret:@"yvtxr3g2njtgoi5"
```

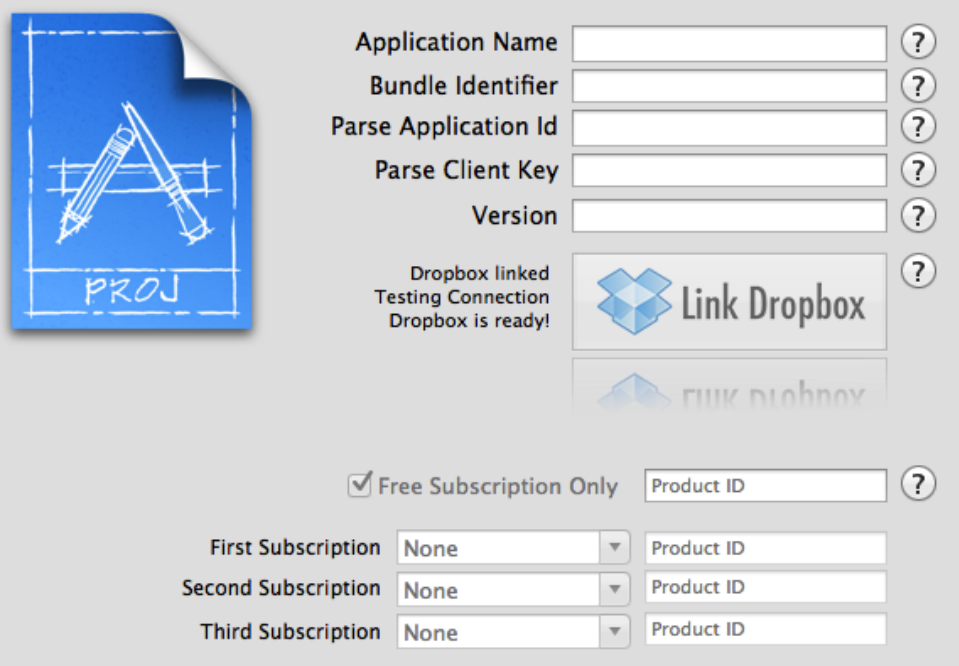
24 Vývojářský portál služby Dropbox je dostupný na adrese https://www.dropbox.com/developers/

```
root:kDBRootDropbox];
[DBSession setSharedSession:dbSession];
```

Výpis 9: Asociace OS X aplikace s uživatelským účtem služby Dropbox

6.3.2 Shromáždění informací a grafických materiálů

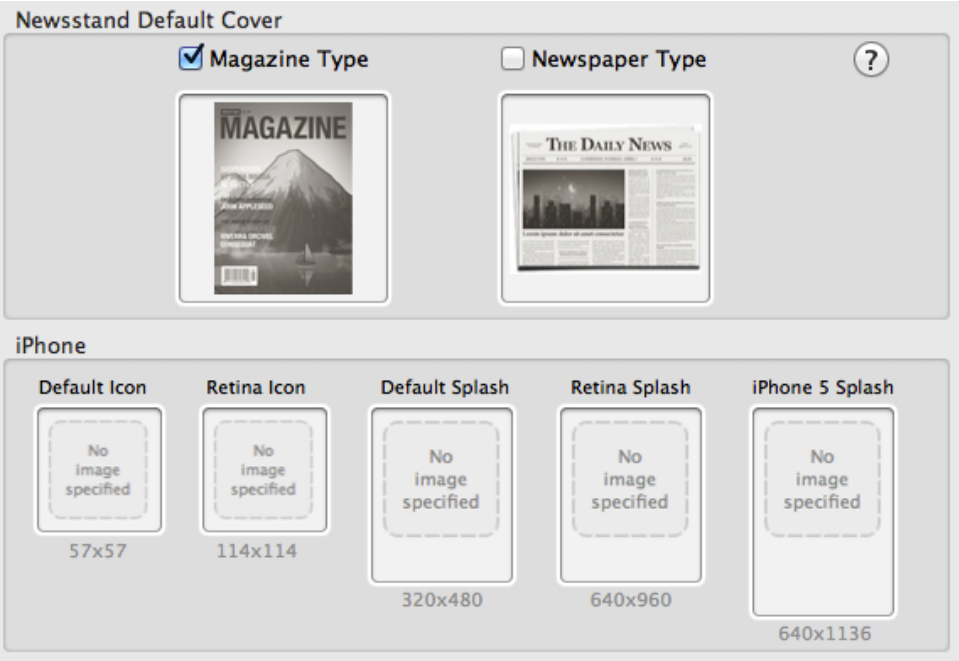
Předtím, než bude vygenerována nová iOS aplikace, je třeba k ní shromáždit informace a grafické materiály. Aplikace implementuje GUI skládající se ze 4 částí, a které uživatele celým procesem postupně provedou. Každá část vyžaduje odlišné vstupní data od uživatele a zároveň zahrnuje jejich validaci.



Obrázek 14: Úvodní část vytváření nové iOS aplikace

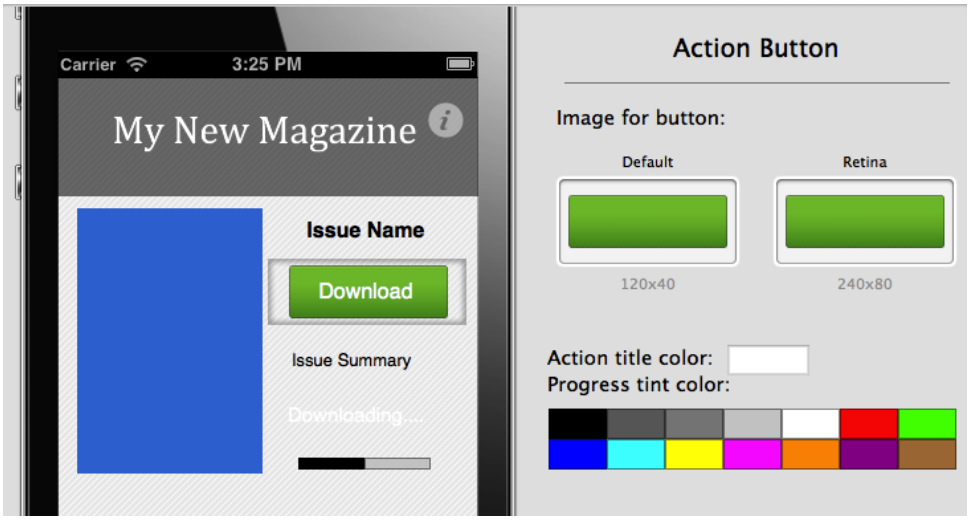
První část je zaměřena na věcné informace o iOS aplikaci a poskytuje přístup k asociování uživatelského účtu služby Dropbox. Uživatel kromě názvu, ID nebo verze aplikace, uvádí klíče pro přístup k API notifikačního serveru Parse a zároveň definuje jaký druh předplatného bude iOS aplikace nabízet.

Druhá část slouží pro vkládání obrázků reprezentující ikony a přední stranu publikace, a to pro různá rozlišení a orientaci mobilního zařízení.



Obrázek 15: Druhá část vytváření nové iOS aplikace

Třetí a čtvrtá část umožňují uživateli měnit vzhled samotné aplikace, a to například vložit vlastní logo, definovat barvu textů nebo pozadí tlačítek.



Obrázek 16: Třetí část vytváření nové iOS aplikace

V momentě, kdy má aplikace všechna potřebná data, lokálně je uloží a začne generovat novou iOS aplikaci.

6.3.3 Vygenerování nové iOS aplikace

Proces generování nové aplikace je více-méně přímočarý. Za účelem generování je implementovaná třída `AppGenerator`. Její instanci jsou předány všechny nezbytné vstupní data, jejichž seznam a struktura je uvedena v Příloze 3. Následně je vytvořena dočasná složka, do které je přesunut archív obsahující zdrojové kódy šablony iOS aplikace. Když je archív rozbalen, jsou do šablony zapsány nové data a zkopírovány všechny použité grafické materiály.

Obrázek 17: Proces generování nové iOS aplikace

Posledním bodem generování je kompilace zdrojových kódů šablony iOS aplikace a vytvoření archívu `.xcarchive`, jež uchová uvnitř všechny soubory a umožní distribuci aplikace na App Store. Samotnou kompilaci zdrojových kódů realizuje rozhraní programu Xcode, které je voláno za pomocí příkazu `xcodebuild`.

```
usr/bin/xcodebuild -scheme MBTemplate -target MBTemplate
-configuration Release archive PRODUCT_NAME=Akademik
```

Výpis 10: Příkaz pro kompilaci aplikace a vygenerování archívu

6.4 Nová publikace

6.4.1 Generování obsahu

Ve chvíli, kdy se uživatel rozhodne vytvořit novou publikaci, OS X aplikace bude požadovat určité vstupní informace, které jsou uvedeny v Tabulce 4.

Vyžadované informace	Popis
Název publikace	Název, pod kterým bude uživateli publikace zobrazena v iOS aplikaci.
Id publikace	Unikátní ID. V případě zpoplatnění publikace by se ID mělo shodovat s ID produktem uvedeného v iTunes Connect.
Popis obsahu	Krátký popis toho, co publikace obsahuje.
Cena	Není-li publikace zdarma, je třeba uvést její cenu, která se shoduje s cenou produktu uvedeného v iTunes Connect.
Datum publikování	Datum kdy byla publikace publikována. Slouží především pro seřazení publikací.
Přední strana publikace	Obrázek přední strany publikace, který bude uživateli zobrazen v iOS aplikaci.
PDF dokument	Slouží jako obsah publikace.

Tabulka 5: Informace vyžadované při generování nové publikace

Po doplnění informací a stisknutí tlačítka pro vytvoření publikace je zahájen proces generování obsahu. PDF dokument je archivován do ZIP souboru a společně s obrázkem přední strany publikace je kopírován na trvalé úložiště. Následně jsou informace zapsány do souboru `Issues.plist`, který uchovává všechny data o vygenerovaných publikacích. Celý proces generování nového obsahu je znázorněn na Obrázku 18.

Obrázek 18: Diagram aktivit generování obsahu nové publikace

6.4.2 Uložení na webový server

Dalším krokem pro distribuci nově vytvořené publikace na mobilní zařízení je uložení informací a obsahu na webový server. Za tímto účelem je implementovaná třída `IssueUploader` využívající metod Dropbox frameworku pro komunikaci s API a je zodpovědná za přesun veškerých souborů. Jakmile se uživatel rozhodne publikovat např. vydání svého nového časopisu, je vytvořena nová instance třídy `IssueUploader` s potřebnými parametry. Instance inicializuje observer pro změny, které nastanou v komunikaci s API služby Dropbox a ověří zda je aplikace asociována s uživatelským účtem[25]. Jestliže ano, `IssueUploader` začne nahrávat data na úložiště.

```
...
// Observer pro Dropbox instanci
[[NSNotificationCenter defaultCenter] addObserver:self
selector:@selector(authHelperStateChangedNotification:)
name:DBAuthHelperOSXStateChangedNotification object:
[DBAuthHelperOSX sharedHelper]];
}

- (void)authHelperStateChangedNotification:(NSNotification
*)notification {
    // vraci, zda je aplikace uz s Dropboxem synchronizovana
    if ([[DBSession sharedSession] isLinked]) {
        NSLog(@"API ONLINE");
        [self uploadCovers];
    }
    else
        NSLog(@"Dropbox is not linked");
}
```

Výpis 11: Připojení k API služby Dropbox

Samotný proces nahrávání dat je mírně komplikovaný. Předtím, než je přesunut jakýkoliv soubor na úložiště, je potřeba ověřit, zda již neexistuje soubor se stejným názvem. V případě, že ano, původní soubor by nebyl automaticky přepsán, ale na úložišti by byly umístěny oba dva soubory a nejnovějšímu souboru by byl pouze pozměněn jeho název. Informace o publikaci ale počítají s původním názvem čímž by došlo k tomu, že dřívější soubor by nebyl nikdy přepsán a iOS aplikace by nikdy nezískala přístup k novému souboru. Instance tedy nejdříve volá metodu `restClient:loadedMetadata:`, která vrací referenci na již existující soubor. Teprve pak lze zahájit nahrávání nového souboru.

```
DBMetadata *childRev = nil;
for (DBMetadata *child in metadata.contents) {
    NSString *folderName = [[child.path pathComponents]
lastObject];
    if ([folderName isEqualToString:file]) {
           childRev = child; break;
```

25 Asociování aplikace s účtem služby Dropbox je již podrobněji popsáno v kaptiole 6.3.1

```
        }
}
NSString *filePath = [NSString stringWithFormat:@"%@/%@",[self
getLocalIssuePath],file];
[[self restClient] uploadFile:file toPath:[self
getDropboxIssuePath] withParentRev:childRev.rev
fromPath:filePath];
```

Výpis 12: Nahrání souboru na úložiště

Jakmile jsou na úložiště Dropboxu úspěšně nahrány všechny soubory a je aktualizován `Store.plist`, uchovávající veškeré informace o distribuovaných publikacích, nastává vhodný okamžik pro upozornění iOS aplikace, že je k dispozici nové vydání.

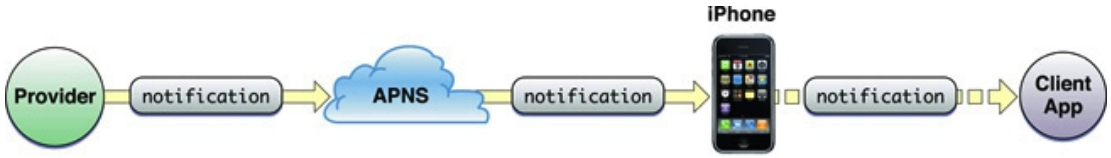
6.4.3 Zasílání notifikací

Pro zasílání notifikací na mobilní zařízení poskytuje společnost Apple speciální službu Apple Push Notification service. Jakým způsobem se iOS aplikace registruje k této službě a jak zpracovává přijatou notifikaci bylo již popsáno v kapitole 5.5.5. APNs nedisponuje rozhraním, které by přímo umožňovalo zasílat notifikace na mobilní zařízení a je tedy nezbytné využít serveru, který naváže spojení s APNs. Za tímto účelem je využíváno notifikační serveru Parse. Ve chvíli, kdy OS X aplikace chce upozornit na novou publikaci, zašle žádost na API notifikačního serveru s nezbytnými parametry viz Tabulka 6.

Parametr	**Popis**
X-Parse-Application-Id	Unikátní ID, které specifikuje definovanou aplikace v uživatelském profilu na portálu Parse.
X-Parse-REST-API-Key	Klíč řídící úroveň práv aplikace.
Obsah zprávy	JSON objekt zasílaný na APNs.
Adresa API	Adresa na kterou je zaslána žádost.

Tabulka 6: Parametry nezbytné pro zaslání žádosti na API Parse

Notifikační server spravuje seznam speciálních klíčů mobilních zařízení, které se při inicializaci jednotlivých iOS aplikací k serveru autentizovali. Server naváže spojení s APNs a předá ji ID aplikace, které je notifikace určena. Následně začne APNs zasílat data, jež obsahují speciální klíče mobilních zařízení a informace, které se mají doručit aplikaci. APNs pomocí klíčů rozliší, kterým zařízením má notifikace rozeslat [12].



Obrázek 19: Proces zasílání notifikací z notifikačního serveru

6.5 Testování

I jednotlivé části OS X aplikace byly testovány především v samotném průběhu jejich vývoje. K testování bylo využito dvou zařízení MacBook Pro a MacBook Air, jejichž specifikace se nachází v Tabulce 7. Zvolená zařízení jsou velice podobná a mají v tomto případě k dispozici více než přebytečné množství zdrojů.

Komponenta	**Sestava použitá při testování**	**Sestava použitá při testování**
OS	OS X 10.8.3	OS X 10.8.3
CPU	2.3 GHz Intel Core i5	1.7 GHz Inter Core i5
RAM	8 GB 1333 MHz DD3	4GB 1333 MHz DD3
Úložiště	160GB SSD, HFS+, zápis 200 MB/s	128GB SSD, HFS+, zápis 183 MB/s

Tabulka 7: Specifikace sestav použitých při testování OS X aplikace

Při generování nové iOS aplikace dochází ke kompilaci a kopírování většího množství souborů. Jelikož se testovací zařízení nepatrně liší v rychlosti zápisu dat, doba pro generování nové aplikace se různí. Jedná se avšak pouze o zanedbatelný rozdíl, který se nedá implementací ovlivnit.

Jeden z požadavků na realizaci vlastního řešení je, aby generovaná iOS aplikace splňovala podmínky pro distribuci společnosti Apple (viz kapitola 3.2). Jediný způsob, jak ověřit zda vygenerovaná aplikace splňuje všechny podmínky a lze ji tak distribuovat skrze App Store, je pokusit se aplikaci vydat. Po 14 dnech od zaslání aplikace na App Store přišla od týmu společnosti Apple kladná odpověď. E-mail od týmu společnosti Apple z 21.8. 2012 „*The status for the following app has changed to Processing for App Store...*“

7 Závěr

Cílem bakalářské práce bylo navrhnout vlastní řešení elektronického publikování jehož součástí bude technologie automatizovaného generování obsahu publikace a znovupoužitelné API pro vývoj iOS aplikací. V předchozích kapitolách byl proveden návrh a popis takovéhoto řešení. Následně byly implementovány dvě ukázkové aplikace realizující distribuci publikací koncovým uživatelům a generování jejich obsahu. V současné chvíli je celé řešení zcela připraveno k použití a splňuje vymezené požadavky. Pokud jde o zhodnocení výsledků testování, pak lze říci, že v rámci základních požadavků jsou dostačující.

Obě ukázkové aplikace nabízí mnoho prostorou pro rozšíření. V případě iOS aplikace by bylo vhodné nahradit nebo upravit některé metody realizující stahování tak, aby aplikace byla schopna stahovat i více publikací najednou nebo aby byla schopna stahování pozastavit a následně obnovit v bodě, kde byla právě přerušena. Dále by iOS aplikace měla implementovat analytické nástroje, které by shromažďovaly podrobnější data o jejím užívání. Žádoucí by mohly být také informace v jakých lokalitách je aplikace používána, a které publikace jsou uživateli více stahovány. Aby řešení bylo více kompaktní, shromažďované informace by byly dostupné přímo v OS X aplikaci, která by realizovala rozhraní pro jejich správu. OS X aplikace by navíc měla poskytnout prostředí pro vkládání různých interaktivních objektů do obsahu publikace. Mezi tyto objekty by určitě měli patřit odkazy na webové stránky, videa, zvuky nebo animace. To však závisí i na frameworku, který je zodpovědný za zobrazování obsahu publikace v iOS aplikaci.

Realizované řešení bylo využito již v praxi a díky němu byla vydána elektronická verze školního časopisu Akademik. Aplikace časopisu Akademik je dostupná skrze App Store[26] a za uplynulý rok si ji stáhlo už stovky uživatelů.

26 Elektronická verze časopisu Akademik je dostupná na adrese https://itunes.apple.com/cz/app/akademik-magazin/id551978674?mt=8

8 Literatura

[1] - iOS Developer Library: Newsstand FAQ. *iOS Developer Library* [online]. [2012-01-19]. Dostupné z: <https://developer.apple.com/library/ios/#technotes/tn2280/_index.html>

[2] - KRILL, Paul. Native mobile app dev vs. HTML5: Why not both? *InfoWorld* [online]. [2012-11-02]. Dostupné z: <http://www.infoworld.com/t/mobile-development/native-mobile-app-dev-vs-html5-why-not-both-206167>

[3] - POKORNÝ, Lukáš. Formáty elektronických knih: specifika a popularita. *Inflow: information journal* [online]. 2012, roč. 5, č. 9 [cit. 2013-04-11]. Dostupné z: <http://www.inflow.cz/formaty-elektronickych-knih-jejich-specifika-popularita>. ISSN 1802-9736.

[4] - iOS Developer Library: Subscriptions. *App Store Resourse Center* [online]. Dostupné z: <https://developer.apple.com/appstore/in-app-purchase/subscriptions.html>

[5] - Mac Developer Library: What Are Cocoa Bindings? *iOS Developer Library* [online]. [2009-03-08]. Dostupné z: <https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/CocoaBindings/Concepts/WhatAreBindings.html#//apple_ref/doc/uid/20002372-CJBEJBHH>

[6] - Mac Developer Library: Cocoa Design Patterns. *Mac Developer Library* [online]. [2010-12-13]. Dostupné z: <https://developer.apple.com/library/mac/#documentation/cocoa/conceptual/CocoaFundamentals/CocoaDesignPatterns/CocoaDesignPatterns.html>

[7] - Mac Developer Library: About Property Lists? *Mac Developer Library* [online]. [2010-03-04]. Dostupné z: <https://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/PropertyLists/AboutPropertyLists/AboutPropertyLists.html#//apple_ref/doc/uid/10000048i-CH3-SW2>

[8] - Mac Developer Library: Transitioning to ARC Release Notes? *Mac Developer Library* [online]. [2012-07-17]. Dostupné z: <http://developer.apple.com/library/mac/#releasenotes/ObjectiveC/RN-TransitioningToARC/Introduction/Introduction.html>

[9] - iOS Developer Library: Newsstand Kit Framework Reference. *iOS Developer Library* [online]. [2011-10-12]. Dostupné z: <http://developer.apple.com/library/ios/#documentation/StoreKit/Reference/NewsstandKit_Framework/_index.html#//apple_ref/doc/uid/TP40010838>

[10] - Mac Developer Library: Local and Push Notification Programming Guide. *Mac Developer Library* [online]. [2011-08-09]. Dostupné z: <http://developer.apple.com/library/mac/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/IPhoneOSClientImp/IPhoneOSClientImp.html>

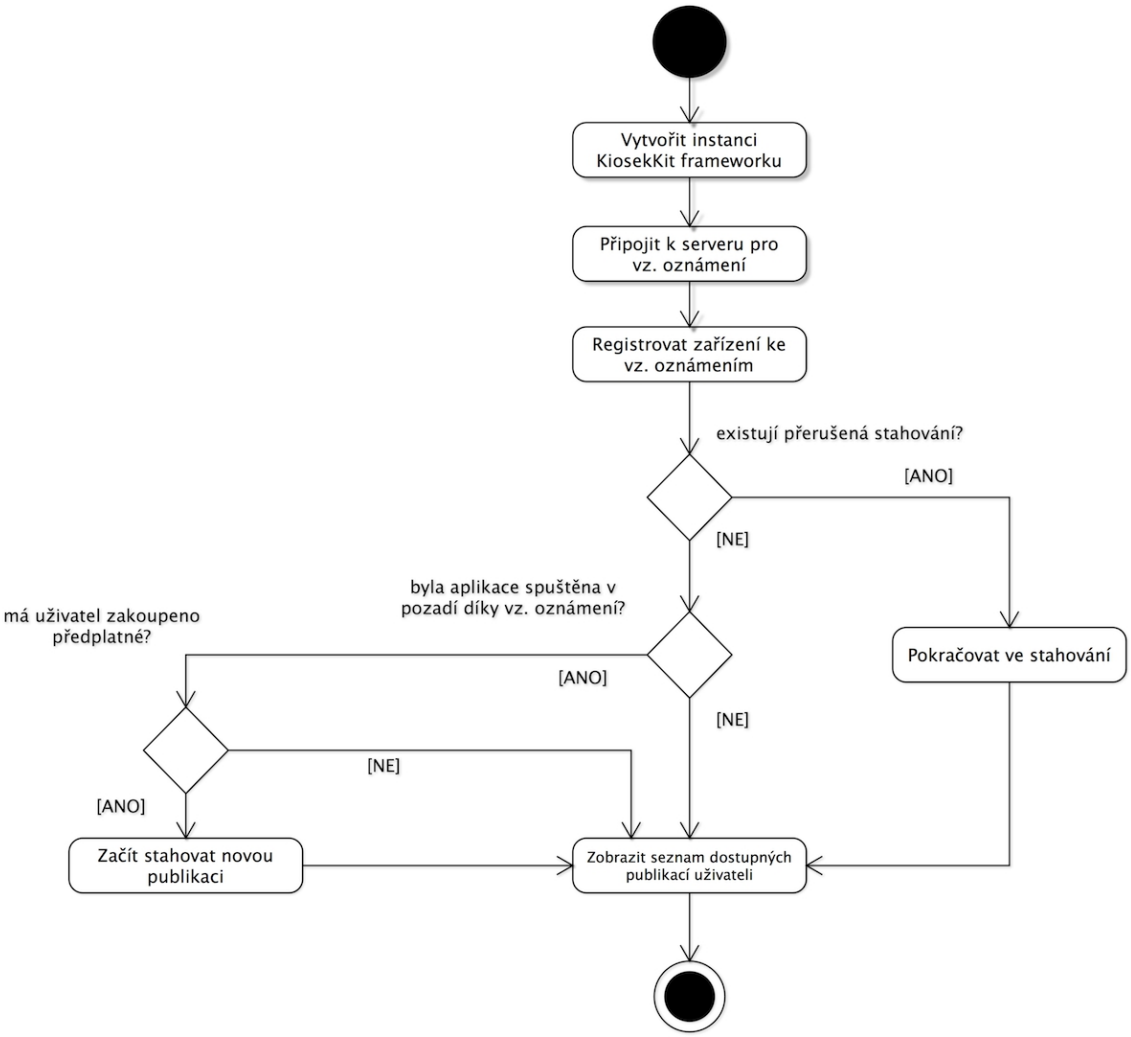
[11] - iOS Developer Library: In-App Purchase Programming Guide. *iOS Developer Library* [online]. [2012-09-19]. Dostupné z:

<http://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/StoreKitGuide/APIOverview/OverviewoftheStoreKitAPI.html>

[12] - iOS Developer Library: Local and Push Notification Programming Guide. *iOS Developer Library* [online]. [2011-08-09]. Dostupné z: <http://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html#//apple_ref/doc/uid/TP40008194-CH100-SW9>

9 Seznam příloh

I. Diagram aktivit inicializace iOS aplikace



Obrázek 20: Diagram aktivit inicializace iOS aplikace

II. Popis API KiosekKit frameworku

```
// notifikace zmeny stavu Kiosku
#define notification_store_status_changed @"StoreStatusChanged"

// preddefinovany protokol, ktery zada o instanci ctecky obsahu publikace
@protocol KiosekContentSource <NSObject>
@required
- (UIViewController *) loadContentReaderInstance:(NSURL*)kPath;
@end

// Kiosek.h je singleton trida
@interface Kiosek : NSObject

// hodnoty stavu Kiosku
typedef enum {
    // Kiosek neni inicializovan
    StoreStatusNotInizialized,
    // Kiosek stahuje data
    StoreStatusDownloading,
    // Kiosek stahl vsechny data a vyckava
    StoreStatusReady,
    // Kiosek doslo k chybe pri stahovani dat
    StoreStatusError
} StoreStatusType;

// aktualni hodnota stavu Kiosku
@property (nonatomic,assign) StoreStatusType status;
// delegat
@property (nonatomic,weak) id delegate;
// protocol pro ziskani instance ctecky obsahu publikace
@property (nonatomic,weak) id <KiosekContentSource> readerSource;
// casopis, ktery byl inicializovan pri startu - pres Push Notifikace nebo kvuli
nedokoncenemu stahovani
@property (nonatomic, strong) NSString *assetIssue;
// instance managera In-App nakupu
@property (nonatomic, strong) InAppPurchaseManager *inApp;
// vraci instanci Kiosku - singleton trida
+(id)sharedInstance;
// metoda pro odstraneni casopisu ze zarizeni
-(void) archiveIssue:(NSString*)_id;
// byl vybran casopis v UI
```

```
-(UIViewController*) issueSelected:(Issue *)anIssue;
// vraci casopis ze seznamu na zaklade jeho indexu - [index v poli storeIssues]
-(Issue *)issueAtIndex:(NSInteger)index;
// vraci casopis podle jeho ID - [id casopisu]
-(Issue *)issueWithID:(NSString *)issueID;
// Kiosek zacne stahovat potrebna data v hlavnim vlakne
-(void)startup;
// Kiosek aktualizuje z hostingu data na pozadi
-(void)updateStore;
// vraci pocet casopisu v Kiosku
-(NSInteger)numberOfStoreIssues;
// vraci zda je Kiosek pripraven a aktualizovan
-(BOOL)isStoreReady;
// metoda pro dalsi praci se stazenymi daty - nebo v pripade, ze data nebyla vubec
stazena
- (void)downloadFinished:(NSArray *)storeArray;
```

III. Vstupní data pro generování iOS aplikace

Název parametru	Typ parametru	Popis
APP_NAME	string	název generované aplikace
BUILD_FOLDER	string	cesta k adresáři, kde se nachází šablona aplikace
BUILD_VERSION	string	verze generované aplikace
BUNDLE_ID	string	inikátní identifikátor aplikace
Default	string	úvodní obrázek pro iPhone
Default-568h@2x	string	úvodní obrázek pro iPhone
Default-Landscape@2x~ipad	string	úvodní obrázek pro iPad
Default-Landscape~ipad	string	úvodní obrázek pro iPad
Default-Portrait@2x~ipad	string	úvodní obrázek pro iPad
Default-Portrait~ipad	string	úvodní obrázek pro iPad
Default@2x	string	úvodní obrázek pro iPhone
FREE_SUBSCRIPTION	string	typ předplatného
HOSTING_URL	string	adresa k souboru store.plist uchovávající informace o publikacách
UINewsstandBindingEdge	string	zarovnání přední strany publikace v Newsstandu
UINewsstandBindingType	string	způsob zobrazení přední strany publikace v Newsstandu
ipad	string	obrázek pro ikonu iPadu
ipad@2x	string	obrázek pro ikonu iPadu
iphone	string	obrázek pro ikonu iPhonu
iphone@2x	string	obrázek pro ikonu iPhonu
newsstand_cover	string	obecný obrázek publikace pro Newsstand

Tabulka 8: Vstupní data pro generování nové iOS aplikace